



Licenciatura Engenharia Informática e Multimédia
Instituto Superior de Engenharia de Lisboa
Ano letivo 2022/2023

Modelação e Programação
Relatório: Trabalho Prático 4 - Parte B

Turma: 22D

Nome: Miguel Alcobia

Número: 50746

Professor: João Ventura

Data: 15 de Junho de 2023

Índice

Introdução	1
Objetivos	1
Explicação das classes	4
Novas das funções	6
Idealização da interface	7
Exemplo de funcionamento	9

Introdução

Neste trabalho, os alunos devem demonstrar que aprenderam e dominam a matéria lecionada durante o semestre na unidade curricular de Modelação e Programação, ao realizar um projeto que pode ir desde uma aplicação de gestão até um jogo.

Tendo realizado o programa em consola na primeira parte do TP4 , agora o foco será na interface gráfica.

Objetivos

Tomei como objetivo desenvolver uma interface para a aplicação de gestão de um hospital, realizada na Parte A. Onde os utilizadores seriam divididos em:

- Administradores
- Médicos
- Enfermeiros
- Doentes

A aplicação começara com um login e depois terá vários menus de acesso, consoante a categoria do utilizador que efetuou o login. O Administrador terá mais opções e terá acesso a dados que outros não terão. Além de registar, apagar, listar estes utilizadores, será possível marcar, registar e apagar consultas. Consultas estas que terão como parâmetros os utilizadores referidos acima, além de dados próprios, tal como data, hora e descrição.

Esta segunda versão além da interface, conta com novidades como a verificação da formatação da data, do horário das consultas e conta também com a opção de alterar a password. Estas novidades serão abordadas mais à frente.

Esta aplicação teria o seguinte UML (próximas duas páginas). Pela grande complexidade do UML, este foi dividido em duas páginas e, ainda assim, não se verá muito bem no documento; contudo estará inserido no *zip* juntamente do presente PDF.

Figura 1 - Primeira Parte do UML

Figura 2 - Segunda Parte do UML

Explicação das classes

User - Classe que representa os users e da qual vão estender as classes que representam o Médico, Enfermeiro e Doente e Admin. Não foi criada como abstrata, pois no decorrer no código são criados objetos do tipo User. Nesta classe existe um método selectCreateElem, que consoante a categoria do user, chama o createElement correspondente.

Admin - Classe que representa os admins e que estende do User. No seu construtor já está definida a sua categoria. Nesta classe existe o método createElement para criar um elemento no XML, representando o Admin e um build que constrói um Admin a partir dos dados do XML.

Médico- Classe que representa os médicos e que estende do User. No seu construtor já está definida a sua categoria e tem como atributo próprio a especialidade do médico. Nesta classe existe o método createElement para criar um elemento no XML, representando o Médico e um build que constrói um Médico a partir dos dados do XML.

Enfermeiro- Classe que representa os enfermeiros e que estende do User. No seu construtor já está definida a sua categoria e tem como atributo próprio a área onde o enfermeiro atua. Nesta classe existe o método createElement para criar um elemento no XML, representando o Enfermeiro e um build que constrói um Enfermeiro a partir dos dados do XML.

Doente- Classe que representa os doentes e que estende do User. No seu construtor já está definida a sua categoria e tem como atributo próprio o diagnóstico. Nesta classe existe o método createElement para criar um elemento no XML, representando o Doente e um build que constrói um Doente a partir dos dados do XML.

Login - Classe onde tem toda a lógica para se fazer login e se proceder à sua respetiva validação. Também tem o método para logout.

Sistema - Classe que estende do Login para que o sistema só funcione se o login for bem sucedido. Nesta classe estão os modos de operação para cada categoria de user, pois dependendo dela têm-se acesso a diferentes opções. Além disso é aqui que estão os métodos para os menus de registar, apagar e listar os diferentes users e consultas funcionarem corretamente.

Consulta - Classe que representa uma consulta, com todos os devidos dados no construtor. Nesta classe existe o método createElement para criar um elemento no XML, representando a Consulta e um build que constrói uma Consulta a partir dos dados do XML.

CmdSistema - Classe onde estão os prints dos vários menus que serão impressos na consola. Além disso, existe um método Scanner para realizar scanners e otimizar tempo durante a realização do código; por fim, um método lerOpts que lê a opção escolhida pelo user.

SistemaBD - Classe que simula uma base de dados, já com alguma informação de base e com métodos para funcionamento da base dados como: apagar, procurar, verificar se existem Users e Consultas. Além destes, ainda existem métodos merge para juntar a informação já existente com a do XML e outros de “save” para atualizar dados da base.

App - Método principal da aplicação, onde se inicia realmente o funcionamento da aplicação em si.

InterApp - Método principal da interface, onde se inicia a *frame* e todos *panels* que serão apresentadas ao utilizador.

LoginPanel - Classe que, assim como todos os *panels*, estende da class JPanel do JSwing. Constrói o painel de Login e apresenta o botão para fazer login e um para alterar a palavra-passe.

MenuAdminPanel - Classe que constrói o painel de Menu Administrador e apresenta um botão para as diversas atividades permitidas a esta categoria de user.

MenuMedPanel - Classe que constrói o painel de Menu Médico e apresenta um botão para as diversas atividades permitidas a esta categoria de user.

MenuEnfPanel - Classe que constrói o painel de Menu Enfermeiro e apresenta um botão para as diversas atividades permitidas a esta categoria de user.

MenuDntPanel - Classe que constrói o painel de Menu Doente e apresenta um botão para as diversas atividades permitidas a esta categoria de user.

RegistAdminPanel - Classe que constrói o painel de Registo para o Administrador e apresenta um campo para cada dado do user a ser registado.

RegistMedPanel - Classe que constrói o painel de Registo para o Médico e apresenta um campo para cada dado do user a ser registado.

RegistEnfPanel - Classe que constrói o painel de Registo para o Enfermeiro e apresenta um campo para cada dado do user a ser registado.

RegistDntPanel - Classe que constrói o painel de Registo para o Doente e apresenta um campo para cada dado do user a ser registado.

MarcarConsltPanel - Classe que constrói o painel de Marcar Consultas e apresenta um campo para cada dado da consulta a ser registada. Neste painel são implementadas os novos métodos de verificação da data e hora.

DeleteAdminPanel - Classe que constrói o painel de Apagar um Administrador e apresenta um campo para cada dado do user a ser eliminado.

DeleteMedPanel - Classe que constrói o painel de Apagar um Médico e apresenta um campo para cada dado do user a ser eliminado.

DeleteEnfPanel - Classe que constrói o painel de Apagar um Enfermeiro e apresenta um campo para cada dado do user a ser eliminado.

DeleteDntPanel - Classe que constrói o painel de Apagar um Doente e apresenta um campo para cada dado do user a ser eliminado.

DeleteConsltPanel - Classe que constrói o painel de Apagar uma Consulta e apresenta um campo para cada dado da consulta a ser eliminada.

ListaAdminsPanel - Classe que constrói o painel de Administradores Registados e apresenta uma tabela com os respetivos users.

ListaMedsPanel - Classe que constrói o painel de Médicos Registados e apresenta uma tabela com os respetivos users.

ListaEnfsPanel - Classe que constrói o painel de Enfermeiros Registados e apresenta uma tabela com

os respetivos users.

ListaDntsPanel - Classe que constrói o painel de Doente Registados e apresenta uma tabela com os respetivos users.

ListaConstlsPanel - Classe que constrói o painel de Consultas e apresenta uma tabela com os diversas consultas marcadas.

Novas Funções

Check Hora: Função da classe Sistema que serve para verificar se a hora respeita o formato **hh:mm** e se nenhum horário passa das 22h00.

Check Data: Função da classe Sistema que serve para verificar se a data respeita o formato **dd/mm/aaaa**.

A opção de alterar a password não é um método, apenas se atribui a password “*default*” a um novo user que foi registado. Podendo este alterar a sua palavra-passe.

```
/**
 * Método para veirficar o formato da data
 * @param data Data inserida pelo user
 * @return true ou false consoante a data respeite o formato de data definido
 */
1 usage
public static boolean checkData(String data){

    if (data.length() != 10 || (data.charAt(2)!='/'||data.charAt(5)!='/')){
        JOptionPane.showMessageDialog( parentComponent: null, message: "Respeite o formato da data [dd/mm/aaaa]");
        return false;
    }else{
        int day = Integer.parseInt(data.substring(0, 2));
        int month = Integer.parseInt(data.substring(3, 5));
        int year = Integer.parseInt(data.substring( beginIndex: 6));

        if ((day < 1 || day > 31) || (month < 1 || month > 12) || (year < 1)) {
            JOptionPane.showMessageDialog( parentComponent: null, message: "Data inválida");
            return false;
        } else {
            return true;
        }
    }
}
```

Figura 3 - Função checkData


```

/**
 * Método para verificar o formato da Hora
 * @param hora Hora inserida pelo user
 * @return true ou false consoante a hora respeite o formato de hora definido
 */
1 usage
public static boolean checkHora(String hora){

    if (hora.length() != 5 || hora.charAt(2)!=':'){
        JOptionPane.showMessageDialog( parentComponent: null, message: "Respeite o formato da hora [hh:mm]");
        return false;
    }else{
        int hour = Integer.parseInt(hora.substring(0, 2));
        int min = Integer.parseInt(hora.substring( beginIndex: 3));
        if (hour>=22 || hour<8 && min>=0){
            JOptionPane.showMessageDialog( parentComponent: null, message: "Só são marcadas consultas entre as 08:00 e as 22:00");
            return false;
        }else {
            return true;
        }
    }
}
}

```

Figura 4 - Função checkHora

Idealização da interface

Com a ajuda do editor de imagem online Canva, idealizou-se a interface gráfica para aplicação. Procurou-se algo minimalista e uma paleta de cores que remetesse para a medicina, saúde etc. (ignorando nesta etapa a preocupação pelo tipo de letra). Alguns dos resultados dos esboços foram os seguintes:



Figura 5 - Mockup do menu do Admin



Figura 6 - Mockup da página de Admin registrados

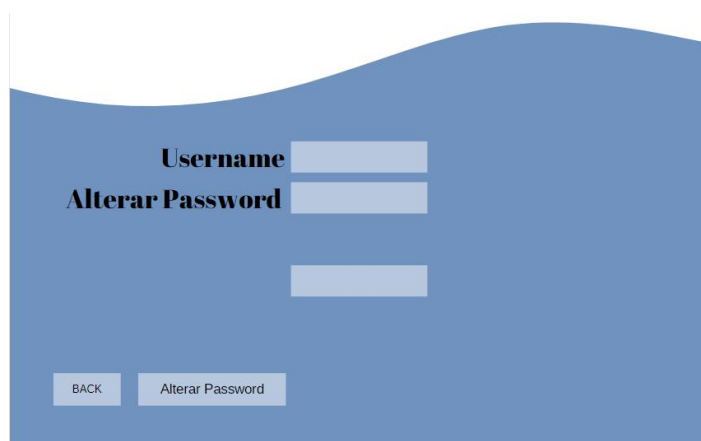


Figura 7 - Mockup da página de login

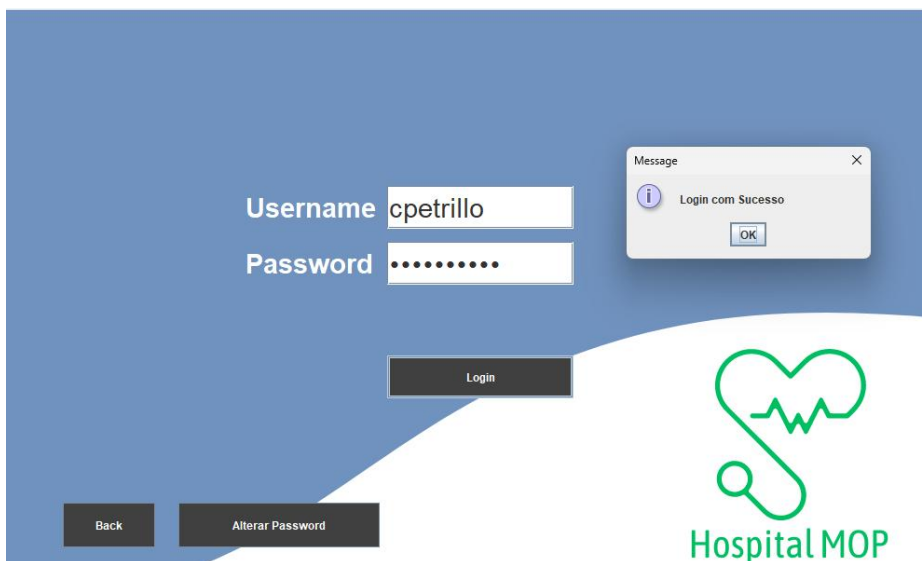
Exemplo de funcionamento

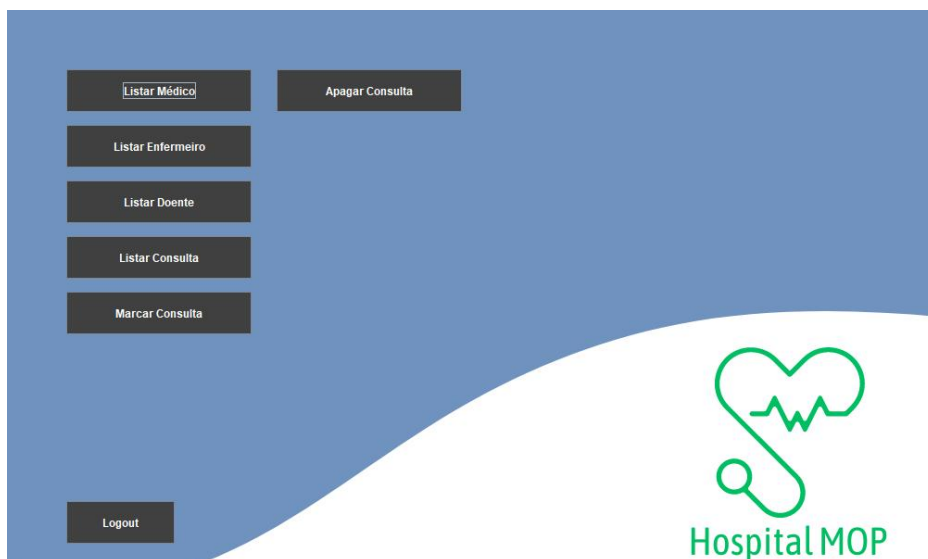
Para exemplificar o funcionamento da app, vamos assumir o contexto apresentado na Parte A.

Vamos assumir o papel da doutora Chiara Petrillo e marcar uma consulta para o Carlos Ferreira:
Os dados são:

```
<medico>
  <nome>Chiara Petrillo</nome>
  <idade>28</idade>
  <code>M13612</code>
  <user>cpetrillo</user>
  <pass>tMtVvQY8BP</pass>
  <especialidade>Nutrição</especialidade>
  <categoria>medico</categoria>
</medico>
```

Procedemos então ao login e ao ser bem sucedido, é apresentado o MENU MÉDICO:





Marcamos então a consulta:

MARCAR CONSULTA

Paciente	Carlos Ferreira
Médico	Chiara Petrillo
Descrição	Consulta de R
Data	23/05/2023
Hora	09:00
Código (CXXXXX)	C02827

Message ×

i Consulta registado com sucesso

OK

Insira o seu código

Insira o seu código

OK

Marcar

Vamos verificar se a consulta está realmente marcada:

CONSULTAS



Código a filtrar:

Consultas de: Chiara Petrillo [medico]

CÓDIGO	DATA	HORA	DESCRIÇÃO	DOENTE
C69852	1/12/2023	14:30	Exames aos intestinos	Carlos Ferreira
C02827	23/05/2023	09:00	Consulta de Rotina	Carlos Ferreira

Insira o seu código

Como podemos ver, a consulta foi marcada com sucesso. Agora efetuamos o logout e verificar que os dados foram gravados no XML:



Users salvos no XML com sucesso!